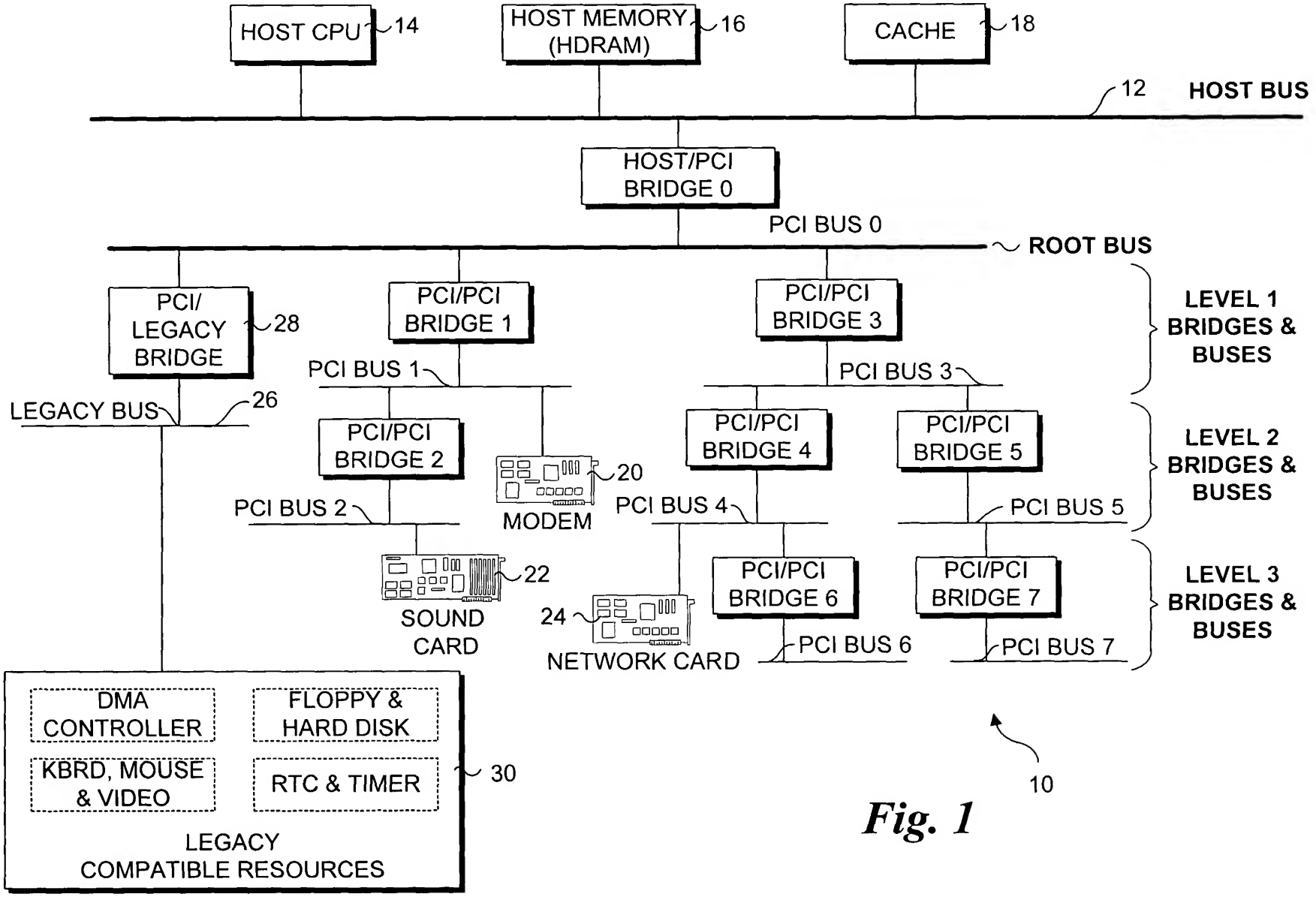
HOST CPU
14
HOST MEMORY (HDRAM)
16
CACHE
18
12
HOST BUS
HOST/PCI BRIDGE 0
PCI BUS 0
ROOT BUS
PCI/ LEGACY BRIDGE
28
PCI/PCI BRIDGE 1
PCI/PCI BRIDGE 3
LEVEL 1 BRIDGES & BUSES
PCI BUS 1
PCI BUS 3
LEGACY BUS
26
PCI/PCI BRIDGE 2
PCI/PCI BRIDGE 4
PCI/PCI BRIDGE 5
LEVEL 2 BRIDGES & BUSES
20
MODEM
PCI BUS 2
PCI BUS 4
PCI BUS 5
22
SOUND CARD
24
NETWORK CARD
PCI/PCI BRIDGE 6
PCI/PCI BRIDGE 7
LEVEL 3 BRIDGES & BUSES
PCI BUS 6
PCI BUS 7
DMA CONTROLLER
FLOPPY & HARD DISK
KBRD, MOUSE & VIDEO
RTC & TIMER
30
LEGACY COMPATIBLE RESOURCES
10

Fig. 1

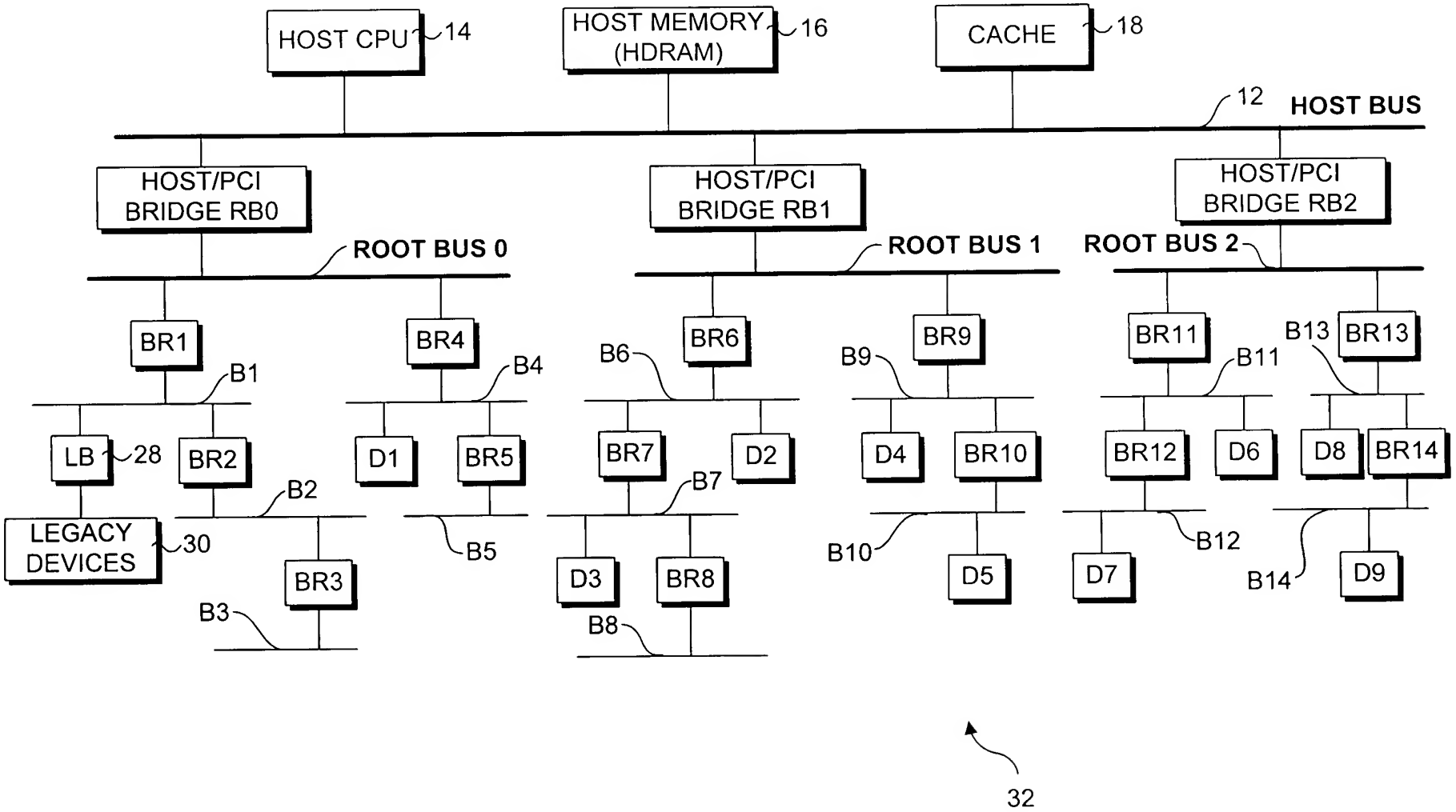
HOST CPU
14
HOST MEMORY (HDRAM)
16
CACHE
18
12
HOST BUS
HOST/PCI BRIDGE RB0
HOST/PCI BRIDGE RB1
HOST/PCI BRIDGE RB2
ROOT BUS 0
ROOT BUS 1
ROOT BUS 2
BR1
B1
BR4
B4
B6
BR6
B9
BR9
BR11
B11
B13
BR13
LB
28
BR2
B2
D1
BR5
BR7
B7
D2
D4
BR10
BR12
D6
D8
BR14
LEGACY DEVICES
30
B5
B10
B12
B14
BR3
B3
D3
BR8
B8
D5
D7
D9
32

Fig. 2

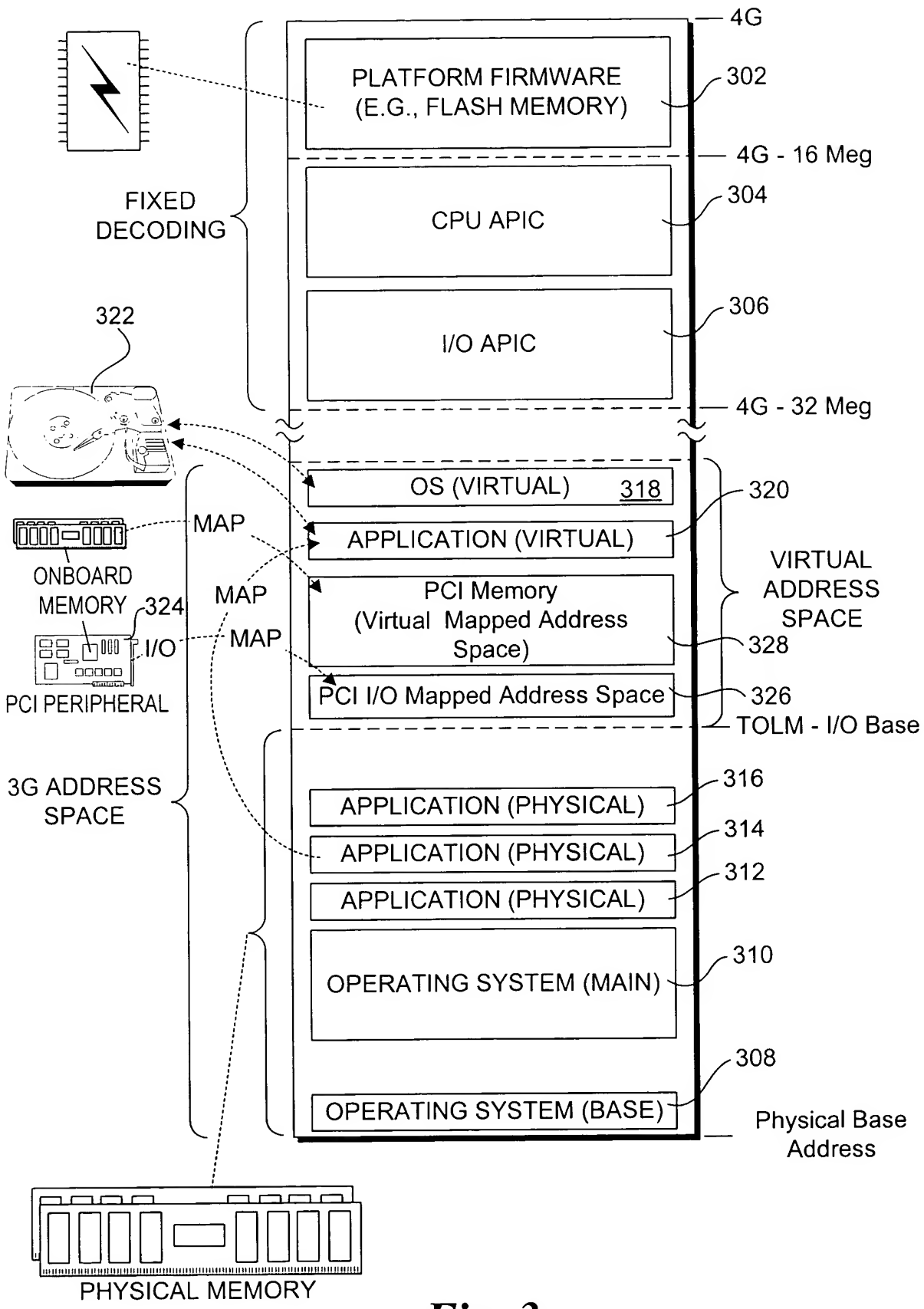
4G
PLATFORM FIRMWARE
(E.G., FLASH MEMORY)
302
4G - 16 Meg
FIXED
DECODING
CPU APIC
304
I/O APIC
306
322
4G - 32 Meg
OS (VIRTUAL) 318
320
MAP
APPLICATION (VIRTUAL)
ONBOARD
MEMORY
VIRTUAL
ADDRESS
SPACE
MAP
324
PCI Memory
(Virtual Mapped Address
Space)
328
MAP
I/O
PCI I/O Mapped Address Space
326
PCI PERIPHERAL
TOLM - I/O Base
3G ADDRESS
SPACE
316
APPLICATION (PHYSICAL)
314
APPLICATION (PHYSICAL)
312
APPLICATION (PHYSICAL)
OPERATING SYSTEM (MAIN)
310
308
OPERATING SYSTEM (BASE)
Physical Base
Address
PHYSICAL MEMORY

Fig. 3

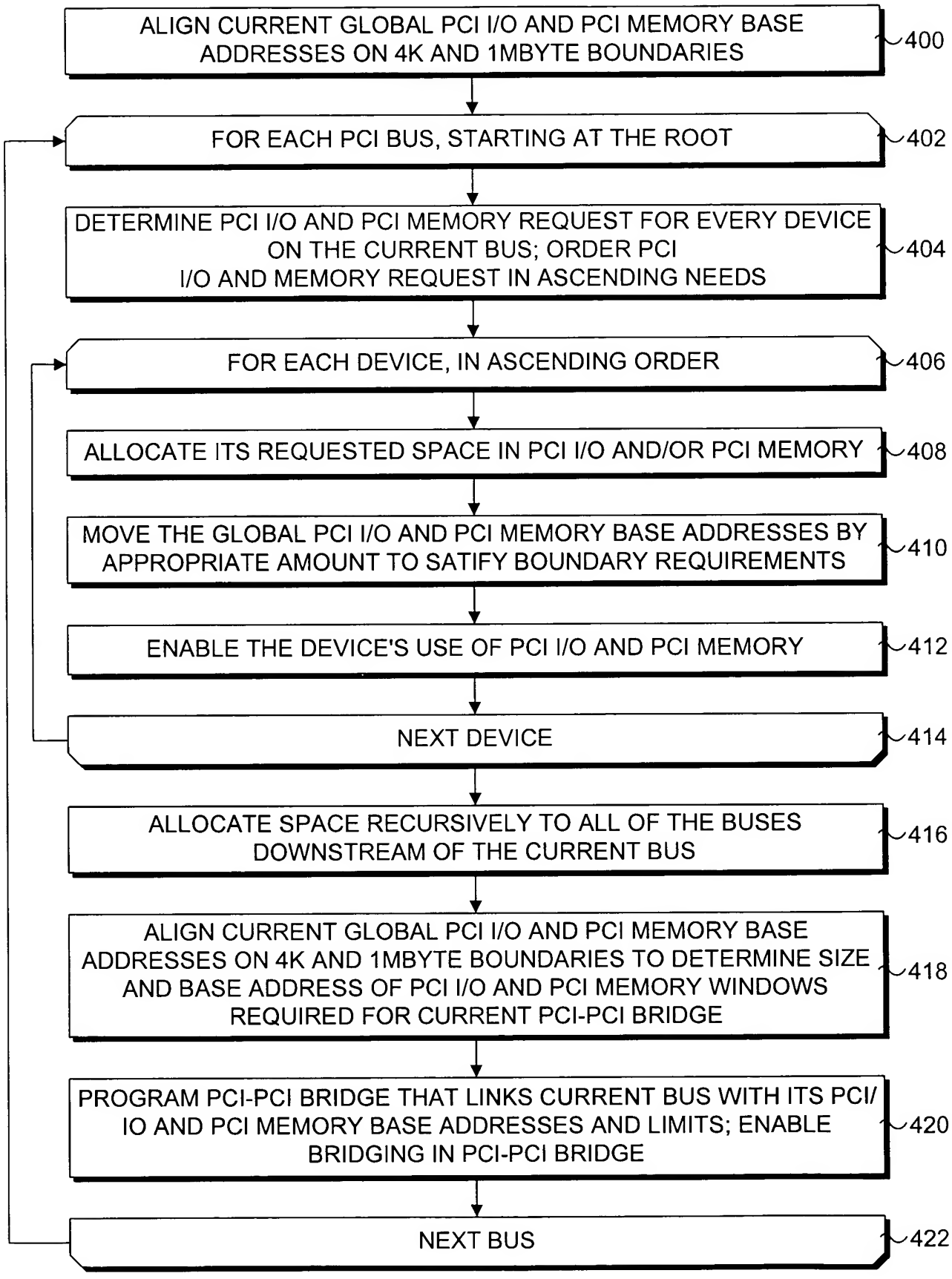

Fig. 4 (Prior Art)

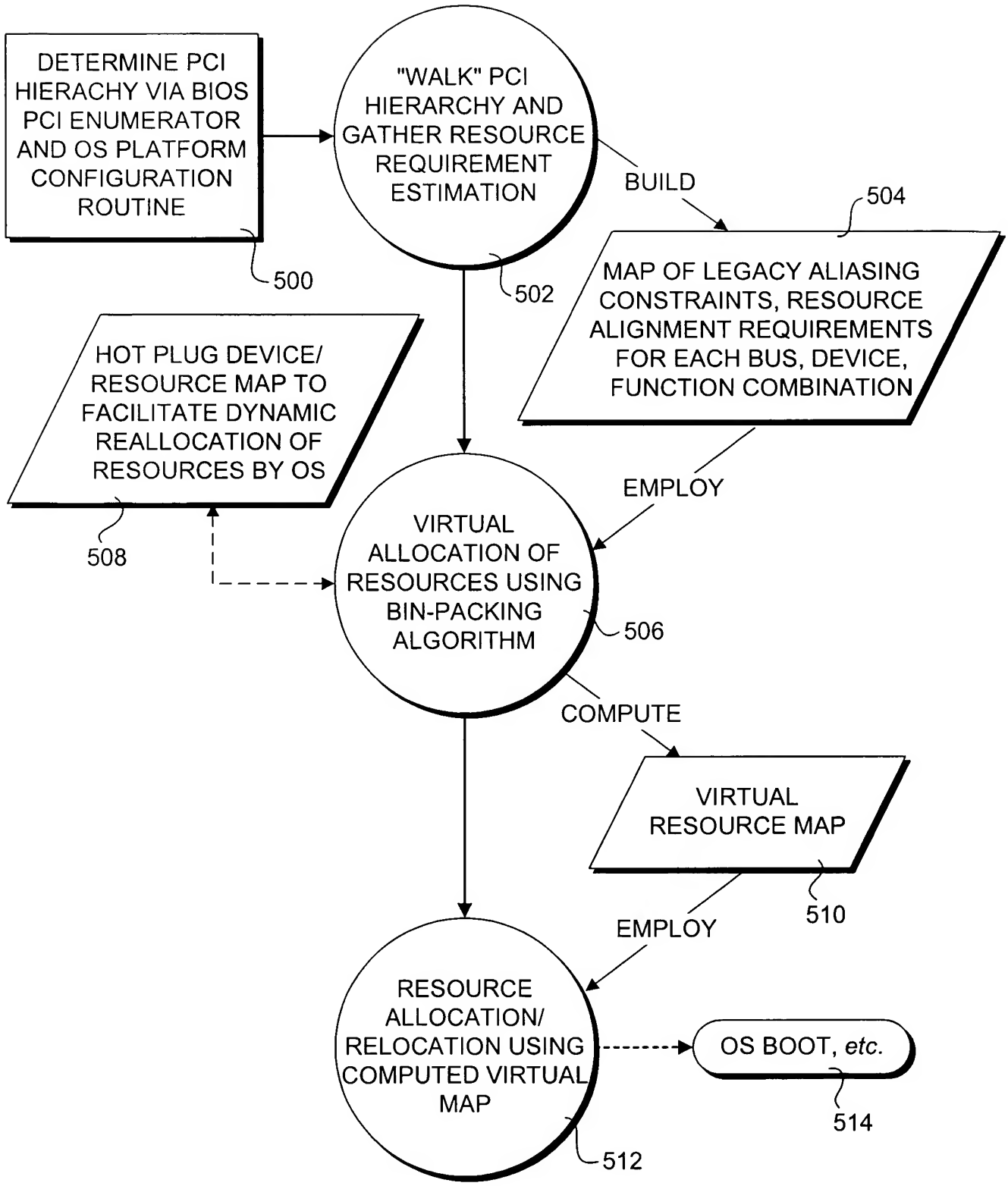
DETERMINE PCI HIERACHY VIA BIOS PCI ENUMERATOR AND OS PLATFORM CONFIGURATION ROUTINE
500
"WALK" PCI HIERARCHY AND GATHER RESOURCE REQUIREMENT ESTIMATION
502
BUILD
504
MAP OF LEGACY ALIASING CONSTRAINTS, RESOURCE ALIGNMENT REQUIREMENTS FOR EACH BUS, DEVICE, FUNCTION COMBINATION
HOT PLUG DEVICE/ RESOURCE MAP TO FACILITATE DYNAMIC REALLOCATION OF RESOURCES BY OS
508
EMPLOY
VIRTUAL ALLOCATION OF RESOURCES USING BIN-PACKING ALGORITHM
506
COMPUTE
VIRTUAL RESOURCE MAP
510
EMPLOY
RESOURCE ALLOCATION/ RELOCATION USING COMPUTED VIRTUAL MAP
512
OS BOOT, etc.
514

Fig. 5

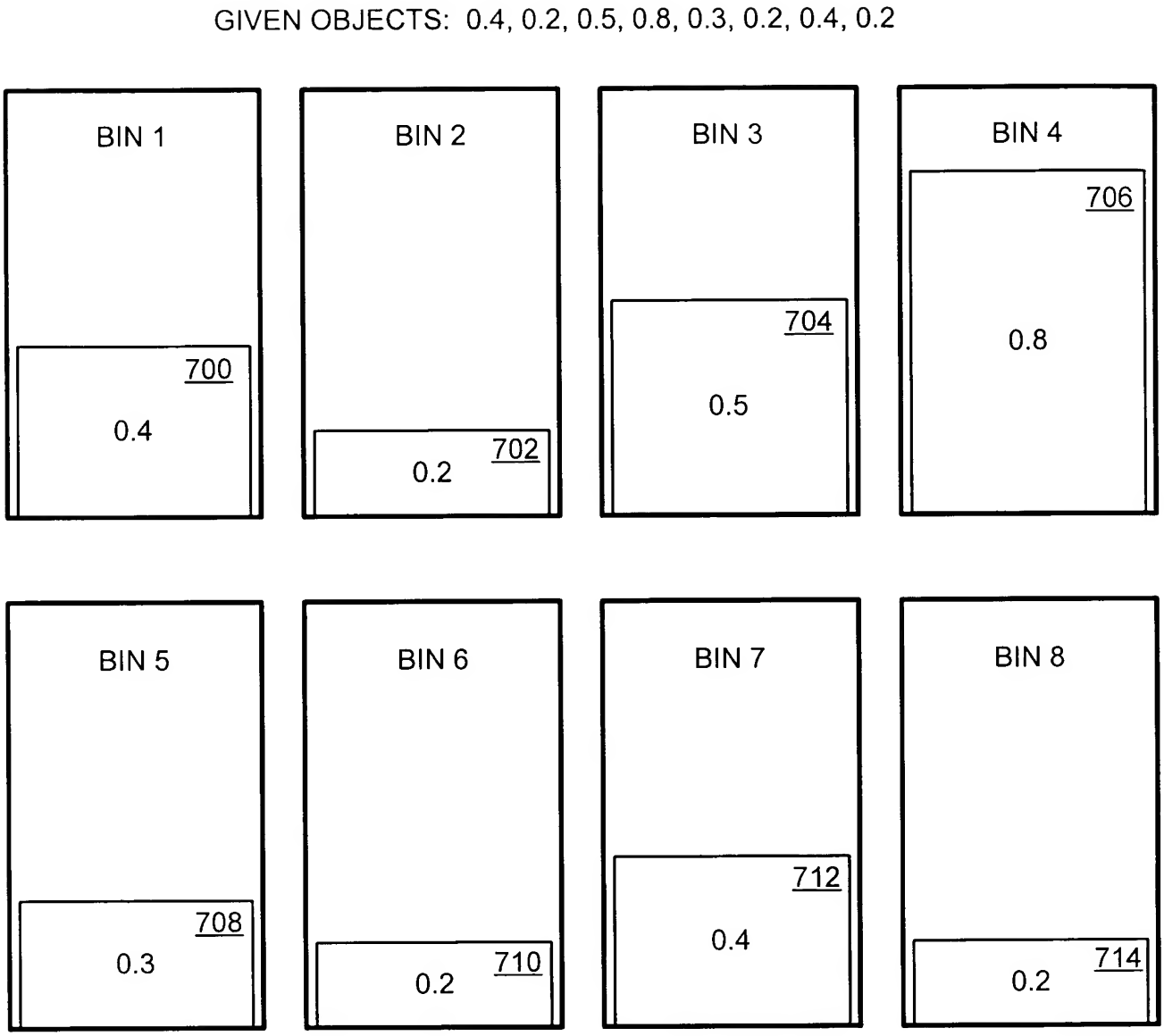

Fig. 6 (Prior Art)

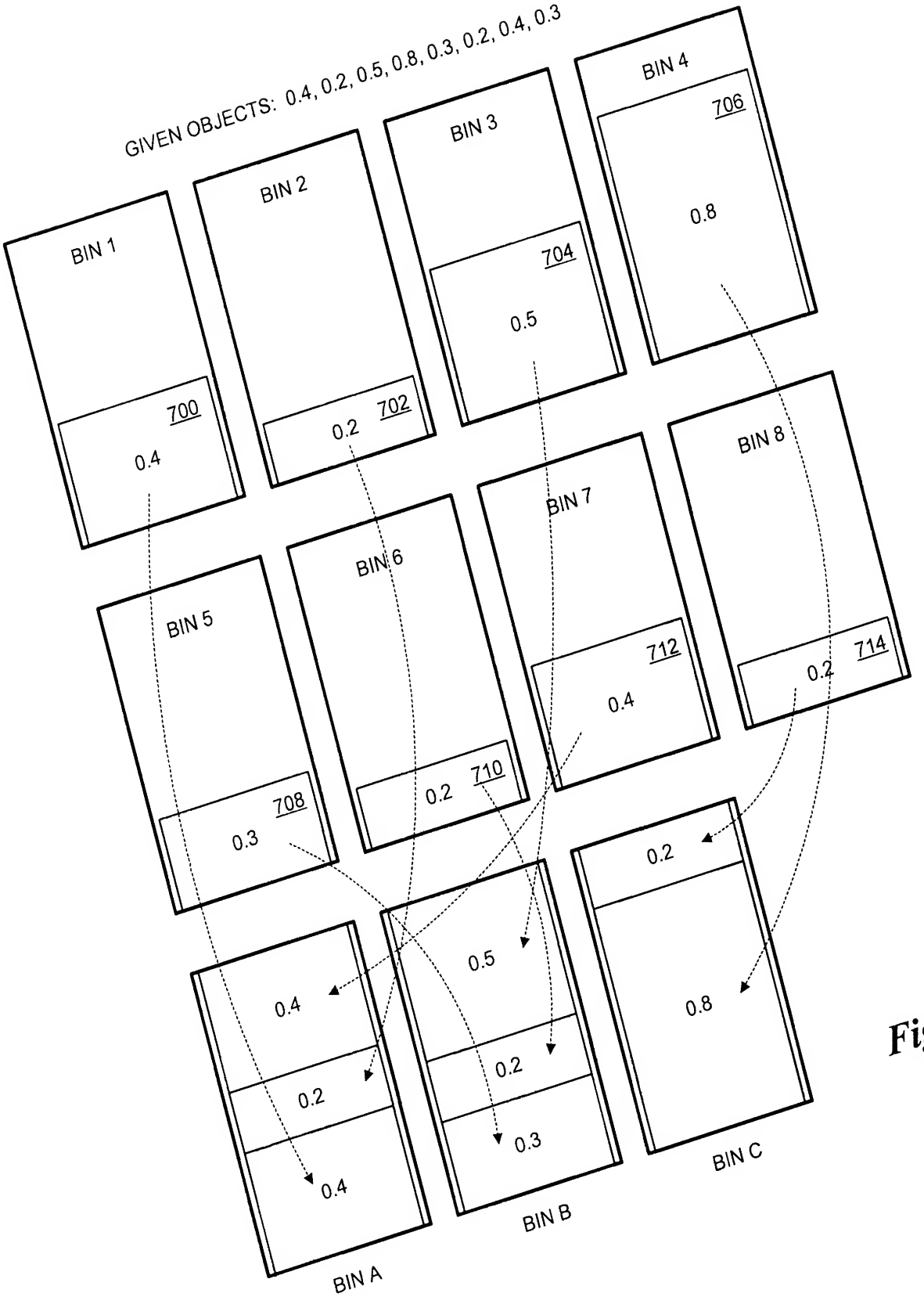
GIVEN OBJECTS: 0.4, 0.2, 0.5, 0.8, 0.3, 0.2, 0.4, 0.3
BIN 1
700
0.4
BIN 2
0.2
702
BIN 3
704
0.5
BIN 4
706
0.8
BIN 5
708
0.3
BIN 6
0.2
710
BIN 7
712
0.4
BIN 8
0.2
714
0.4
0.2
0.4
BIN A
0.5
0.2
0.3
BIN B
0.2
0.8
BIN C

Fig. 7

```
knapSack_k(C,S) // S an array with n values (the sizes)
   int maxSum, sum, j; // C is the size of the knapsack
   Set T = new Set;
   take = ∅; maxSum = 0;
   for each subset T ⊆ {1,...,n}
                        with k elements {
      sum = the sum of the values S[i] such that i ∈ T;
      if (sum ≤ C) {
            for each j not in T {
            if (sum + S[j] ≤ C){
               sum = sum + S[j];
               T = T ∪ {j};}
            }
            if (maxSum < sum) {maxSum = sum}
     }
   }  return maxSum;
```

Fig. 8

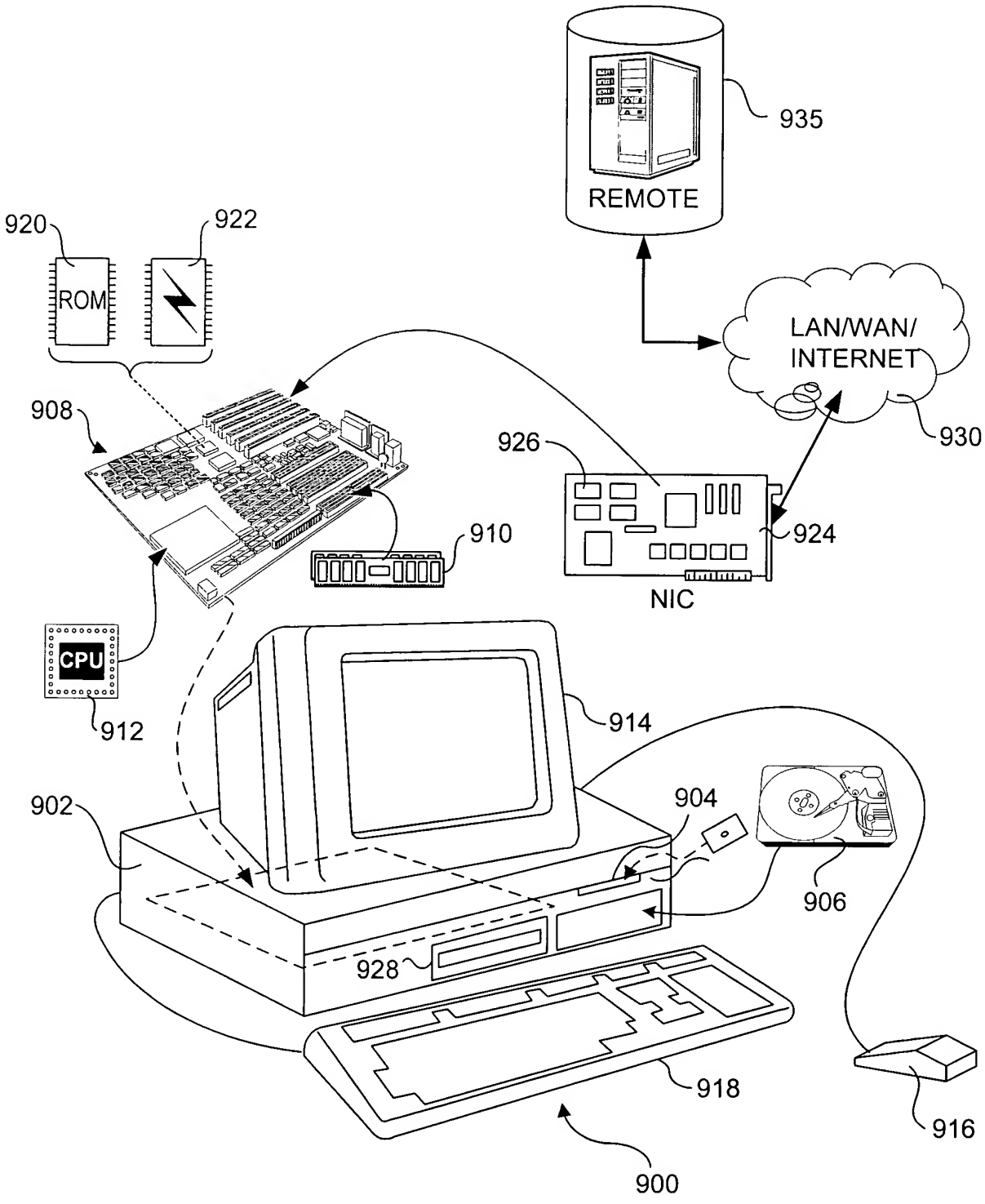
935
REMOTE
920
922
ROM
LAN/WAN/
INTERNET
930
908
926
910
924
NIC
CPU
912
914
902
904
906
928
918
916
900

Fig. 9